Screencast: 13-process-and-resource-management.webm or 13-process-and-resource-management.mp4

LaUSAH REFERENCE - Chapter 5, Controlling Processes
TLCL REFERENCE - Chapter 11, Processes

What you may already know

How do you manage processes on Windows and Mac?

What is the init process?

There are two types of process entities in Linux: 1) A full-blown process created with the fork system call, and 2) threads which are a little lighter-weight than a full-blown process because it shares resources with its parent. Threads are created with the clone system call and the use of a threading library.

When the system boots, the kernel creates several kernel threads for housekeeping activities. Then it starts the init process which has a process ID number of 1. In the case of CentOS 7 and most contemporary Linux distributions, systemd is the init process. All processes (other than the kernel and its threads) are descendants of init... init is the "mother of all processes".

What is a PID?

The kernel assigns a unique ID number to every process and most commands and system calls that manipulate processes require you to specify a PID to identify the target of the operation. PIDs are issued sequentially.

What is process ownership?

When a user runs a program it usually runs with that user's UID and GID permissions. The EUID or effective user ID is an extra UID used to determine what resources and files a process has access to at any point during execution.

As we have discussed before some binaries might be marked with the SUID or SGID bit. A SUID/SGID program may only need elevated permissions for a few system calls and uses the EUID as a method to raise and lower access as needed.

What is a priority and niceness?

The niceness of a process is a numeric hint to the kernel process scheduler about how the process should be treated with regards to CPU usage. The common range of allowable values is -20 (most favorable scheduling) to 19 (least favorable). A child process inherits the niceness value of its parent.

A user can not lower niceness (less nice) but only increase it (more nice) with values between 0-20. The root user can alter niceness in both directions.

The nice command is used to start a program at a specified niceness. The renice command is used to alter niceness.

Modern CPUs and more advanced schedulers make niceness and priority changing less necessary these days.

What is a process status?

D Uninterruptible sleep (usually IO)
R Running or runnable (on run queue)
S Interruptible sleep (waiting for an event to complete)
T Stopped, either by a job control signal or because it is being traced.
W paging (not valid since the 2.6.xx kernel)
X dead (should never be seen)
Z Defunct ("zombie") process, terminated but not reaped by its parent.

(man ps)

What are signals? (man 7 signal or kill -l [list])

Signals are process-level interrupt requests. If a process is working properly and has been written to do certain things upon certain signals it can catch and respond. One common signal is 15 / SIGTERM which is where a program is asked to quit. Another is signal 9 / SIGKILL which is usually used when a program is not responding to 15.

Signals described in the original POSIX.1-1990 standard.

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated

SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at tty
SIGTTIN	21,21,26	Stop	tty input for background process
SIGTTOU	22,22,27	Stop	tty output for background process

Commands you'll want to learn

- ps : auxwww
- pstree : -nup
- kill : -15 is default, -9 is handy
- top : M, P, u
- /proc
- sysstat service : monitors load every 10 minutes
  - sar
- Developers and advanced users should check out strace

Settings that affect user resource usage

/etc/security/limits.conf

```
#Each line describes a limit for a user in the form:
#
#<domain> <type> <item> <value>
#
#Where:
#<domain> can be:
# - an user name
# - a group name, with @group syntax
# - the wildcard *, for default entry
# - the wildcard %, can be also used with %group syntax,
# for maxlogin limit
#
#<type> can have the two values:
# - "soft" for enforcing the soft limits
# - "hard" for enforcing hard limits
#
#<item> can be one of the following:
# - core - limits the core file size (KB)
# - data - max data size (KB)
# - fsize - maximum filesize (KB)
# - memlock - max locked-in-memory address space (KB)
# - nofile - max number of open files
# - rss - max resident set size (KB)
# - stack - max stack size (KB)
# - cpu - max CPU time (MIN)
# - nproc - max number of processes
```

```
# - as - address space limit
# - maxlogins - max number of logins for this user
# - maxsyslogins - max number of logins on the system
# - priority - the priority to run user process with
# - locks - max number of file locks the user can hold
# - sigpending - max number of pending signals
# - msgqueue - max memory used by POSIX message queues (bytes)
#
#<domain> <type> <item> <value>
#
@student hard nproc 75
@student hard memlock 50000
@student hard fsize 500000
@student hard data 50000
@student hard nofile 500
@student hard locks 500
@student hard cpu 60
@student hard memlock 200000
@student hard priority 2
@student hard as 500000
```

Disk Management

Bad things happen when a partition or disk gets full. Therefore it is important that you learn how to identify disk availability problems and resolve them when needed. Two commands that you'll mainly use are:

df: -h and -i are most common flags
du: -s and -h are the most common flags

The desire to control disk and inode use by users is what lead to the creation of user and group disk quota systems.

cgroups and systemd

Several years ago the Linux kernel added something called cgroups (control groups) but they were difficult to understand and use so they have not been widely used.

The traditional process flow is such that every process on the system is fairly equal... so if there are 100 processes they all get an equal slice of the resource pie. With cgroups that model changes and a cgroup becomes an additional unit the scheduler understands. This allows related processes to be grouped/scheduled together making it harder for any process and its children to bog the system down. It also has the added benefit that a cgroup is more easily and reliably killed than a bunch of individual processes.

Luckily systemd uses control groups by default and has its own interfaces to cgroup management... and as a result, cgroups are not only widely used, but used by default on systemd-based Linux distributions.

Here is a 3.5 minute video that shows systemd's cgroup features:

or https://www.youtube.com/watch?v=-25oWssr9WI (included at end of lecture screencast video)

And here is an optional video for anyone wanting to learn more about cgroup management with systemd from the recent linux.conf.au 2017: Managing performance parameters through systemd (YouTube)